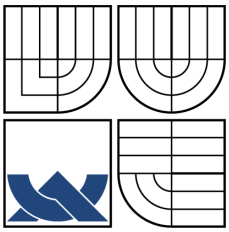
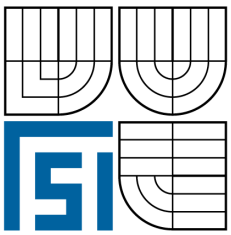


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA STROJNÍHO INŽENÝRSTVÍ
ÚSTAV MATEMATIKY

FACULTY OF MECHANICAL ENGINEERING
INSTITUTE OF MATHEMATICS

GEOMETRICKÉ ALGORITMY V ROBOTICE.

COMPUTATION GEOMETRY IN ROBOTICS.

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

MAREK PIVOVARNÍK

VEDOUCÍ PRÁCE
SUPERVISOR

Mgr. JAROSLAV HRDINA, Ph.D.

BRNO 2010

Vysoké učení technické v Brně, Fakulta strojního inženýrství

Ústav matematiky
Akademický rok: 2009/2010

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

student(ka): Marek Pivovarník

který/která studuje v **bakalářském studijním programu**

obor: **Matematické inženýrství (3901R021)**

Ředitel ústavu Vám v souladu se zákonem č.111/1998 o vysokých školách a se Studijním a zkušebním řádem VUT v Brně určuje následující téma bakalářské práce:

Geometrické algoritmy v robotice.

v anglickém jazyce:

Computation geometry in robotics.

Stručná charakteristika problematiky úkolu:

Plánování pohybu robota pomocí geometrických algoritmů. Uvažujeme dvojrozměrného robota pohybujícího se po rovině se stabilními polygonálními překážkami (robot bude také polygonální). Praktické využití je například při pohybu robota po podlaze.

Cíle bakalářské práce:

Osvojení základních pojmů geometrických algoritmůs jejích aplikace.

Seznam odborné literatury:

Computational Geometry: Algorithms and Applications, Second Edition by Mark de Berg, M. van Krefeld, M. Overmars, and O. Schwarzkopf (Hardcover - Feb 18, 2000)

Vedoucí bakalářské práce: Mgr. Jaroslav Hrdina, Ph.D.

Termín odevzdání bakalářské práce je stanoven časovým plánem akademického roku 2009/2010.

V Brně, dne 27.10.2009

L.S.

prof. RNDr. Josef Šlapal, CSc. Ředitel ústavu	prof. RNDr. Miroslav Doupovec, CSc. Děkan fakulty

Abstrakt

Práca sa zaoberá určením zakázaného pracovného priestoru pre polygonálneho robota. Hlavným cieľom je výpočet tvaru prekážky v konfiguračnom priestore. Tento problém je riešený pomocou Minkowského sumy. Ďalej práca pojednáva o vlastnostiach Minkowského sumy. Zvyšnú časť práce tvorí uvedenie algoritmu pre výpočet Minkowského sumy, jeho ladenie a implementácia do prostredia C#.

Abstract

Thesis deals with finding forbiden configuration space for polygonal robot. The aim is to compute an obstacle shape in configuration space. This problem solves Minkowsky Sum. Also thesis deals with Minkowsky sum properties. In the rest of the thesis is mentioned algorithm to compute Minkowsky sum, its debugging and implementation to the C# enviroment.

kľúčové slová

Minkowského suma

key words

Minkowsky sum

PIVOVARNÍK, M.: *Geometrické algoritmy v robotice*, Brno, Vysoké učení technické v Brně, Fakulta strojního inženýrství, 2010. 32 s. Vedoucí Mgr. Jaroslav Hrdina, Ph.D.

Prehlasujem, že som bakalársku prácu *Geometrické algoritmy v robotice* vypracoval samostatne pod vedením Mgr. Jaroslava Hrdinu, Ph.D. s použitím materiálov uvedených v zozname literatúry.

Marek Pivovarník

S veľkým potešením by som chcel poďakovať svojmu vedúcemu Mgr. Jaroslavovi Hrdinovi, Ph.D., ktorý ma usmerňoval a pomáhal mi dôležitými radami kedykoľvek som to potreboval a taktiež moje poďakovanie patrí všetkým, ktorí mi priamo, alebo nepriamo pomohli.

Marek Pivovarník

Obsah

Úvod

K tomu, aby bolo možné naplánovať pohyb robota, je väčšinou potrebné, aby robot mal nejaké informácie o prostredí, v ktorom sa bude pohybovať. Informácie o prostredí môže poskytnúť plán rozostavenia prekážok, napríklad rozostavenie stojov, alebo iných komponentov vo firme. Pre prípad pohyblivých prekážok, ktoré nie sú zobrazené v pláne rozostavenia prekážok, musí byť robot vybavený senzormi. Takéto prekážky môžu tvoriť ľudia pohybujúci sa vo firme. Pri plánovaní pohybu robota je dôležité nájsť a presne definovať priestor v ktorom sa robot môže pohybovať.

Pre robota, ktorý má tvar polygonu, je dôležité a potrebné prepočítať tvar prekážok tak, aby nedošlo ku kolízii robota s prekážkou. Pri hľadaní tvaru prekážok sa obmedzíme na dvojrozmerný priestor. Tento predpoklad nie je až tak dramatický, ako sa môže zdať. V praxi mnoho krát stačí tento predpoklad k určeniu priestoru, v ktorom sa robot môže pohybovať. Ďalšími zjednodušujúcimi predpokladmi bude skutočnosť, že robot nebude rotovať a rovnako ako prekážky, bude reprezentvoaný konvexným polygonom.

Tieto špecifikácie a ďalšie predpoklady budú presnejšie rozobrané v prvej kapitole. Ďalej v prvej kapitole budú definované základné pojmy, ako napíklad robot, alebo prekážka. Taktiež v tejto kapitole budú ďalšie dôležité pojmy, ako pracovný a konfiguračný priestor a pojmy zakázaný a voľný konfiguračný priestor. Všetky tieto pojmy sa budú využívať v ďalších kapitolách.

Druhá kapitola je venovaná hľadaniu tvaru prekážky v konfiguračnom priestore. Hlavným problémom je zisiť, kedy nastáva kolízia robota s prekážkou. Minkowského suma ponúka aparát pre výpočet tvaru prekážky v konfiguračnom priestore. V tejto kapitole je uvedený príklad analytického výpočtu Minkowského sumy pre konkrétneho robota a prekážky. Poznámka o referenčnom bode poukazuje na funkcie referenčného bodu robota. Ďalej kapitola pojednáva o základných vlastonostiach Minkowského sumy, ako napríklad tvar Minkowksého sumy, konvexnosť, alebo obsah Minkowského sumy. Niektoré z vlastností využíva algoritmus pre výpočet Minkowského sumy. Taktiež tieto vlastnosti môžu byť prostriedkom k lepšiemu pochopeniu a preniknutiu do Minkowského sumy.

V tretej kapitole, nazvanej Algoritmus pre výpočet Minkowského sumy, sa uvádza pseudokód algoritmu, ktorý počíta tvar Minkowského sumy. Na dvoch príkladoch je uvedené fungovanie algoritmu a jeho ladenie. Vyladený pseudoalgoritmus je implementovaný do prosterdia C#. Okrem ukážky kódu metódy, ktorá implementuje už spomínaný pseudokód sa táto kapitola venuje vytvorením pomocných metód, ktoré sú nevyhnutné pre chod programu. Zdrojové kódy týchto metód sú tu taktiež uvedené. V závere tejto kapitoly je uvedený návod na použitie programu pre výpočet Minkowského sumy, ktorý je priložený v prílohe tejto práce.

Kapitola 1

Základné pojmy

Na úvod je podstatné definovať si základné pojmy, s ktorými budeme neskôr pracovať. Taktiež si stanovíme špecifikácie resp. obmedzenia, ktoré budeme uvažovať v celej práci.

Robota a prekážku definujeme nasledovne.

Definícia 1.1. *Označme robota $\mathcal{R}(x, y)$ ako množinu bodov v rovine, ktorá je reprezentovaná konvexným polygonom s referenčným bodom v (x, y).*

Definícia 1.2. *Označme prekážku $\mathcal{P}$ ako množinu bodov v rovine, ktorá je reprezentovaná konvexným polygonom.*

Konvexná množina bodov je taká množina, v ktorej, ak spojíme úsečkou akékoľvek dva body patriace do tejto množiny, tak táto úsečka bude celá ležať v tejto množine. Ďalej v celej práci predpokladáme, že robot sa pohybuje v rovine a taktiež, že nerotuje resp. nedokáže sa natáčať. Napríklad pohyb auta môže slúžiť ako predstava pobybu robota, ktorý sa dokáže natáčať. Potrebujeme ešte definovať pracovný a konfiguračný priestor.

Definícia 1.3. *Pracovným priestorom robota $\mathcal{R}(x, y)$ nazývame priestor, v ktorom sa robot môže pohybovať a kde jeho poloha je definovaná polohou referenčného boda.*

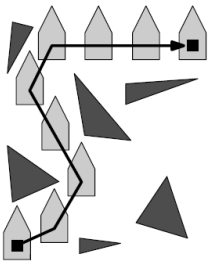
Definícia 1.4. *Konfiguračný priestor robota $\mathcal{R}(x, y)$ je parametrický priestor robota, kde každý jeden bod tohto priestoru definuje presnú polohu robota v pracovnom priestore.*

Pre objasnenie pojmov si uvedieme dva príklady:

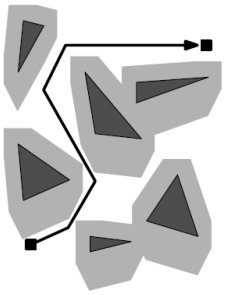
Príklad 1.1
Pre prípad, za vyžšie spomenutých prepokladov, pracovný a konfiguračný priestor je zobrazený na obr. 1.1 a obr. 1.2. Na obr. 1.2 stojí za pozornosť, že nepracujeme s celým polygonálnym robotom, ale iba s jeho referenčným bodom. Môžeme si všimnúť, že ak referenčný bod sa bude nachádzať vo svetlosivej oblasti, tak to znamená, že robot nabúral do prekážky.

Príklad 1.2
Zaujimavejším príkladom bude, ak si prípad s rovnakými predpokladmi ako v príklade 1.1 predstavíme, že robot môže aj rotovať. Pracovný priestor takéhoto robota sa nezmení, no jeho poloha bude definovaná už troma parametrami. Robota by sme mohli označiť takto: $\mathcal{R}(x, y, \varphi)$, kde φ je uhol natočenia robota v protismere hodinových ručiek vzhľadom k osi x.

Konfiguračný priestor už nebude dvojdimenzionálny ale trojdimenzionálny. Naviac, nejedná sa o Euklidovský priestor, ale o priestor definovaný takto: $\mathbb{R}^2 \times \langle 0; 360° \rangle$. Takýto priestor si môžeme predstaviť ako nekonečný valec.



Obr. 1.1: pracovný priestor [1]



Obr. 1.2: konfiguračný priestor [1]

Týmto máme jasne definovanú polohu robota, a to buď zadaním konkrétnych hodnôt parametrov referenčného bodu alebo bodom v konfiguračnom priestore. Nasledujúcimi pojmami rozdelíme konfiguračný priestor na zakázaný a voľný.

Definícia 1.5. *Označme množinu prekážok ako S. Časť konfiguračného priestoru ktorý, obsahuje body odpovedajúce polohám, pri ktorých robot pretína prekážku, nazývame zakázaný konfiguračný priestor. Označme ho ako $\mathcal{C}_{forb}(\mathcal{R}, S)$.*

Definícia 1.6. *Doplnok $\mathcal{C}_{forb}(\mathcal{R}, S)$, ktorý obsahuje body odpovedajúce polohám, pri ktorých robot nepretína prekážky, nazývame voľný konfiguračný priestor. Označme ho ako $\mathcal{C}_{free}(\mathcal{R}, S)$.*

Ak hľadáme cestu pre robota zo štartovacej pozície do cieľovej, tak celá krivka, ktorá predstavuje cestu, musí ležať vo voľnom priestore. V príklade 1.1 je táto krivka znázornená. V totmo príklade na obrázku zobrazujúcom kofiguračný priestor je svetlosivou farbou vyznačný zakázaný konfiguračný priestor. Pre názornosť, jendotlivé prekážky tmavosivej farby zostali zobrazené aj napriek tomu, že už nehrajú žiadnu rolu.

Kapitola 2

Minkowského suma

Situácia je pomerne komplikovaná, pretože budeme musieť zistiť, kedy dochádza k zrážke robota, ako konvexného polygonu, s prekážkou, ktorá je taktiež konvexný polygon. Prekážky v konfiguračnom priestore už nebudú mať rovnaký tvar ako prekážky v pracovnom priestore. V tejto kapitole zavedieme množinu Minkowského suma, ktorá nám vyrieši problém so zrážkami robota s prekážkou a uvedieme jej vlastnosti.

2.1 Definícia Minkowského sumy

Pripomeňme, že máme definovaného robota $\mathcal{R}(x, y)$ a prekážku $\mathcal{P}$. Je zrejmé, že polygonálny robot má rovnaký tvar v pracovnom aj konfiguračnom priestore. V prípade polygonálneho robota má prekážka v pracovnom priestore iný tvar ako v konfiguračnom priestore. Tvar prekážky v konfiguračnom priestore si definujeme nasledovne:

Definícia 2.1. *Uvažujme prekážku $\mathcal{P}$ a robota $\mathcal{R}$ v pracovnom priestore. Označme $\mathcal{CP}$ ako množinu bodov, pre ktorú platí*

$$\mathcal{CP} = \{(x, y) : \mathcal{R}(x, y) \cap \mathcal{P} \neq \emptyset\}.$$

Konvexý obal množiny $\mathcal{CP}$ tvorí tvar prekážky $\mathcal{P}$ v konfiguračnom priestore.

Tvar $\mathcal{CP}$ si môžeme predstaviť ako keby sme robota $\mathcal{R}$ ťahali po okraji prekážky $\mathcal{P}$. Krivka, ktorú vytvorí referenčný bod robota je hranica $\mathcal{CP}$ (viď obr. 2.1).

Prejdime k definícii Minkowského sumy.

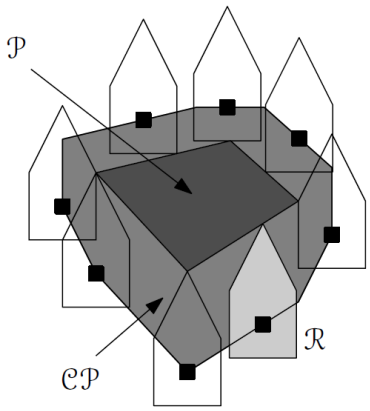
Definícia 2.2. *Majme dve množiny $S_1 \subset \mathbb{R}^2$ a $S_2 \subset \mathbb{R}^2$. Označme Minkowského sumu ako $S_1 \oplus S_2$, ktorá je definovaná ako*

$$S_1 \oplus S_2 = \{p + q : p \in S_1, q \in S_2\},$$

kde $p+q$ označuje vektorovú sumu bodov p a q, čo znamená, ak $p = (p_x, p_y)$ a $q = (q_x, q_y)$, tak potom

$$p + q = (p_x + q_x, p_y + q_y).$$

Nasledujúca veta nás presvedčí o tom, že pomocou Minkowského sumy zavedieme prekážku v konfiguračnom priestore $\mathcal{CP}$. Najskôr, pre bod $p = (p_x, p_y)$ definujeme značenie $-p = (-p_x, -p_y)$, a taktiež pre množinu S definujeme $-S = \{-p : p \in S\}$.



Obr. 2.1: ukážka $\mathcal{CP}$ [1]

Veta 2.1. *Buď $\mathcal{R}$ rovinný robot a $\mathcal{P}$ prekážka. Pre množinu $\mathcal{CP}$ platí*

$$\mathcal{CP} = \mathcal{P} \oplus (-\mathcal{R}(0,0)).$$

Dôkaz. Potrebujeme dokázať, že robot $\mathcal{R}(x,y)$ pretína prekážku $\mathcal{P}$ práve vtedy, ak pre referenčný bod robota platí $(x,y) \in \mathcal{P} \oplus (-\mathcal{R}(0,0))$. Inými slovami, že referenčný bod robota vstúpy do polygonu $\mathcal{CP}$.

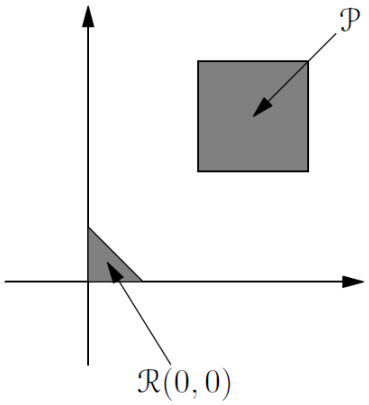
Najprv predpokladáme, že robot pretína prekážku, a teda prepokladáme, že existuje taký bod $q = (q_x, q_y)$, ktorý sa nachádza v priesečniku robota $\mathcal{R}(x,y)$ a prekážky $\mathcal{P}$. Teda platí, že $q \in \mathcal{R}(x,y)$. Z toho vyplýva, že $(q_x - x, q_y - y) \in \mathcal{R}(0,0)$, alebo ekvivaletntný zápis $(-q_x + x, -q_y + y) \in -\mathcal{R}(0,0)$. Pretože súčasne platí, že bod $q \in \mathcal{P}$, tak z toho vyplýva, že $(x,y) \in \mathcal{P} \oplus (-\mathcal{R}(0,0))$

Naopak, buď $(x,y) \in \mathcal{P} \oplus (-\mathcal{R}(0,0))$. Potom existujú body $(r_x, r_y) \in \mathcal{R}(0,0)$ a $(p_x, p_y) \in \mathcal{P}$, pre ktoré platí $(x,y) = (p_x - r_x, p_y - r_y)$, resp. $p_x = r_x + x$ a $p_y = r_y + y$, čo implikuje, že robot $\mathcal{R}(x,y)$ pretína prekážku $\mathcal{P}$. □

Dostali sme alternatívne zavedenie prekážky v konfiguračnom priestore. Touto vetou sme o krok bližšie k vyriešeniu problému zrážok robota $\mathcal{R}(x,y)$ s prekážkou $\mathcal{P}$. Uvedieme si príklad využitia Minkowského sumy, pričom množiny bodov robota a prekážky budú tvoriť iba ich vrcholy.

Príklad 2.1
Prekážku $\mathcal{P} = \{(40,40), (80,40), (80,80), (40,80)\}$ a robota $\mathcal{R}(0,0) = \{(0,0), (0,20), (20,0)\}$ máme zadané. Robot má referenčný bod $(0,0)$. Náčrt je na obr. 2.2. Pre výpočet tvaru prekážky v konfiguračnom priestore aplikujeme vetu 2.1. Prevedieme robota na $-\mathcal{R}(0,0) = \{(0,0), (0,-20), (-20,0)\}$. Každý bod robota $\mathcal{R}(0,0)$ vektorovo sčítame s každým bodom prekážky $\mathcal{P}$ a dostaneme nasledovnú množinu bodov:

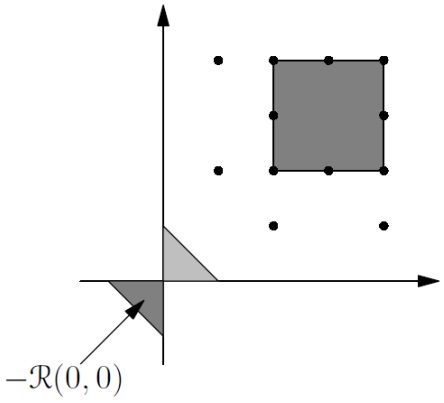
$$\{(40,40), (80,40), (80,80), (40,80), (40,20), (80,20),$$
$$(80,60), (40,60), (20,40), (60,40), (60,80), (20,80)\}.$$



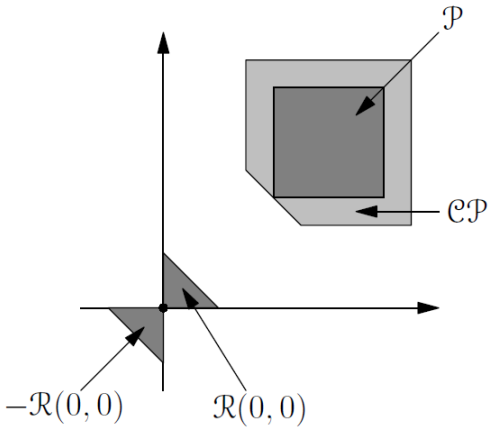
Obr. 2.2: náčrt príkladu

Vytvoríme konvexný obal tejto množiny bodov, čím získavame tvar prekážky $\mathcal{P}$ v konfiguračnom priestore, teda tvar $\mathcal{CP}$. Konvexný obal je tvorený bodmi:

$$\{(40,20),(80,20),(80,80),(20,80),(20,40)\}.$$



Obr. 2.3: výpočet tvaru prekážky $\mathcal{CP}$



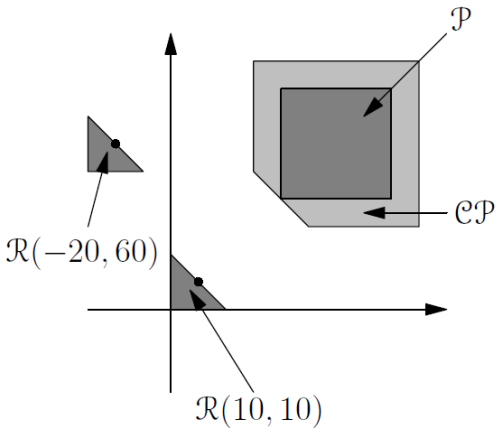
Obr. 2.4: tvar prekážky $\mathcal{CP}$

Poznámka 2.1 Referenčný bod robota má dve funkcie. Prvá funkcia je, že nám udáva polohu robota v pracovnom priestore. Tá druhá nám pomáha pri výpočte Minkowského sumy. Minkowského sumu, a teda tvar prekážok v konfiguračnom priestore, počítame stále pred tým, ako začneme plánovať akúkoľvek cestu robota. Ak máme zadanú polohu robota, teda polohu referenčného bodu robota, je potrebné prepočítať polohu jednotlivých vrcholov polygonu vzhľadom k referenčnému bodu, a až potom pokračovať vo výpočte Minkowského sumy. Prepočet pre každý bod $r = (r_x, r_y)$, patriaci robotovi $\mathcal{R}(x, y)$, voči referenčnému bodu je takýto:

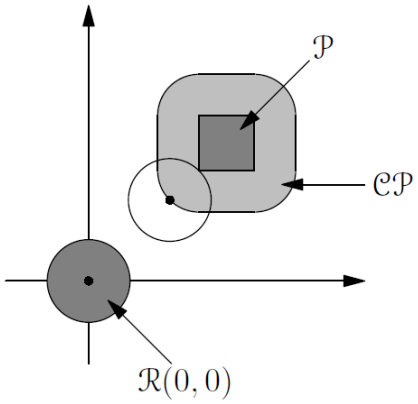
$$(r_x^*, r_y^*) = (r_x, r_y) - (x, y),$$

kde (x, y) predstavuje polohu referenčného bodu robota.

Z poznámky 2.1 vyplýva, že aj keď zmeníme polohu robota v pracovnom priestore, ale poloha referenčného bodu sa voči robotovi nezmení, tak tvar aj poloha prekážky sa v konfiguračnom priestore nezmení. Tento poznatok je zobrazený na obr. 2.5. Pre oba roboty $\mathcal{R}(10,10)$ a $\mathcal{R}(-20,60)$ je tvar aj poloha $\mathcal{CP}$ rovnaká. Touto vlastnosťou sa budeme ešte zaoberať v ďalšej podkapitole.



Obr. 2.5: tvar prekážky pre posunutý refrenčný bod



Obr. 2.6: tvar $\mathcal{CP}$ pre nopolygonálneho robota

Iní príklad tvaru $\mathcal{CP}$, kde robot $\mathcal{R}(0,0)$ je nepolygonálny a prekážka je stále polygon, je zobrazený na obr. 2.6.

2.2 Vlastnosti Minkowského sumy

Uvedieme si niekoľko dôležitých vlastností Minkowského sumy, ktoré neskôr využijeme.

Veta 2.2. *Tvar prekážky $\mathcal{CP}$ nezávisí na polohe referenčného bodu robota.*

Dôkaz. Majme robota $\mathcal{R}(0,0)$, kde $(0,0) \in \mathcal{R}(0,0)$ a prekážku $\mathcal{P}$. Zvoľme $(u,v) \in \mathbb{R}^2$. Ďalej pre robota platí $\mathcal{R}(u,v) = \{(x-u, y-v) : x,y \in \mathcal{R}(0,0)\}$. Je potrebné dokázať nasledujúcu ekvivalenciu:

$$(\xi_1,\xi_2) \in -\mathcal{R}(u,v) \oplus \mathcal{P} \Leftrightarrow (\xi_1,\xi_2) \in -\mathcal{R}(0,0) \oplus \mathcal{P} \oplus (u,v),$$

pričom platí, že $-\mathcal{R}(u,v) \oplus \mathcal{P} \subseteq \mathbb{R}^2$, $-\mathcal{R}(0,0) \oplus \mathcal{P} \oplus (u,v) \subseteq \mathbb{R}^2$ a bod $(\xi_1,\xi_2) \in \mathbb{R}^2$. Dôkaz v smere "$\Rightarrow$"je nasledujúci.

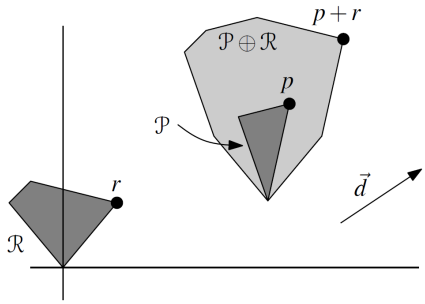
Platí, že $(\xi_1,\xi_2) = -(\varphi_1,\varphi_2) + (p,q)$, pričom $(\varphi_1,\varphi_2) \in \mathcal{R}(u,v)$ a $(p,q) \in \mathcal{P}$. Keďže bod (φ_1,φ_2) je z robota, tak ho môžeme zapísať nasledovne $(\varphi_1,\varphi_2) = (x-u, y-v)$, kde $(x,y) \in \mathcal{R}(0,0)$. Ďalej platí:

$$(\xi_1,\xi_2) = -(x-u, y-v) + (p,q) = -(x,y) + (p,q) + (u,v) \in -\mathcal{R}(0,0) \oplus \mathcal{P} \oplus (u,v),$$

čím je prvá časť dôkazu hotová. Dôkaz v smere "$\Leftarrow$"môžeme považovať za zrejmý. Týmto je veta dokázaná.

□

Veta 2.3. *Majme robota $\mathcal{R}(x,y)$, prekážku $\mathcal{P}$ a $\mathcal{CP} = \mathcal{P} \oplus (-\mathcal{R}(x,y))$. Extrémny bod v smere $\vec{d}$ na $\mathcal{CP}$ je sumou extrémnych bodov v smere $\vec{d}$ na $\mathcal{R}(x,y)$ a $\mathcal{P}$.*



Obr. 2.7: zobrazenie vlastnosti z vety 2.3 [1]

Dôkaz. Ako prvé dokážeme, že extrémny bod Minkowského sumy v smere osi x je sumou extrémnych bodov robota a prekážky v tomto smere. Extrémny bod prekážky a robota v tomto smere nájdeme jednoducho. Sú to body, ktoré majú najväčšiu $x-$ovú hodnotu. Extrémny bod robota $\mathcal{R}(x,y)$ označíme $r = (r_x, r_y)$ a podobne extrémny bod prekážky $\mathcal{P}$ označíme $p = (p_x, p_y)$. Z definície Minkowského sumy pre tieto dva body platí $p \oplus r = e = (p_x + r_x, p_y + r_y)$, a teda získavame nový bod, ktorý označíme e. Z tejto rovnosti je vidieť, že bod e má taktiež maximálnu možnú x–ovú hodnotu, pretože vzniká súčtom maximálnych x–ových hodnôt robota a prekážky. Akýkoľvek iný súčet bude vždy menší.

Pre každý iný smer dokážeme otočiť súradný systém tak, aby smer osi x sa zhodoval s daným smerom. Prerátame súradnice všetkých bodov pre pootočený súradný systém. Podobne, aj v tomto novom súradnom systéme dokážeme nájsť extrémne body robota a prekážky v smere novej osi x. Rovnakým pustupom dostaneme extrémny bod aj pre Minkowského sumu. Z toho vyplýva, že v akomkoľvek smere extrémny bod Minkowského sumy je sumou extrémnych bodov robota a prekážky, čím je veta dokázaná.

□

Veta 2.4. *Majme robota $\mathcal{R}(x,y)$ a prekážku $\mathcal{P}$ s n a m hranami. Konvexný obal Minkowského sumy $\mathcal{CP} = \mathcal{P} \oplus (-\mathcal{R}(x,y))$ má tvar konvexného polygonu s najviac $n+m$ hranami.*

Dôkaz. Na začiatok je potrebné dokázať, že Minkowského suma dvoch konvexných polygonov je konvexný polygon. Budeme vychádzať z definície konvexnosti, ktorá vraví, že úsečka spájajúca akékoľvek dva body musí patriť do polygonu. Predpokládajme body, pre ktoré platí $(\varphi_1, \varphi_2), (\xi_1, \xi_2) \in \mathcal{P} \oplus (-\mathcal{R}(x,y))$. Tieto body spojíme, čím vytvoríme usečku, ktorú zapíšeme parametricky nasledovne:

$$(\varphi_1, \varphi_2) + \rho \cdot (\xi_1 - \varphi_1, \xi_2 - \varphi_2), \quad t \in \langle 0, 1 \rangle. \tag{2.1}$$

Ďalej predpokládajme, že body $(a_1, a_2), (b_1, b_2) \in \mathcal{P}$ a $(c_1, c_2), (d_1, d_2) \in (-\mathcal{R}(x,y))$. Aj v tomto prípade tieto body v jednotlivých polygonoch spojíme a vytovríme usečky, ktorých parametrický zápis je nasledovný:

$$\begin{aligned} &(a_1, a_2) + t \cdot (b_1 - a_1, b_2 - a_2) \subseteq \mathcal{P} \quad ; && t \in \langle 0, 1 \rangle \\ &(c_1, c_2) + s \cdot (d_1 - c_1, d_2 - c_2) \subseteq (-\mathcal{R}(x,y)) \quad ; && s \in \langle 0, 1 \rangle. \end{aligned}$$

Ďalej pre body $(\varphi_1, \varphi_2), (\xi_1, \xi_2)$ uvažujme:

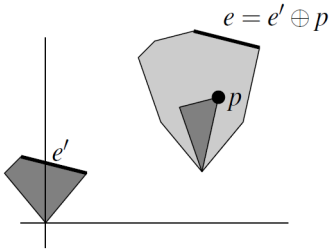
$$\begin{aligned} (\varphi_1, \varphi_2) &= (a_1, a_2) \oplus (c_1, c_2) = (a_1 + c_1, a_2 + c_2) \\ (\xi_1, \xi_2) &= (b_1, b_2) \oplus (d_1, d_2) = (b_1 + d_1, b_2 + d_2) \end{aligned}$$

Dosadením týchto vzťahov do parametrického vyjadrenia úsečky 2.1 dostávame nasledujúce vyjadrenie úsečky, ktoré ďalej upravíme:

$$(a_1 + c_1, a_2 + c_2) + \rho \cdot (b_1 + d_1 - a_1 - c_1, b_2 + d_2 - a_2 - c_2) =$$
$$= (a_1, a_2) + \rho \cdot (b_1 - a_1, b_2 - a_2) + (c_1, c_2) + \rho \cdot (d_1 - c_1, d_2 - c_2) \subseteq (\mathcal{P} \oplus (-\mathcal{R}(x, y))),$$

kde $\rho \in \langle 0, 1 \rangle$. Teda z toho vyplýva, že úsečka spájajúca akékoľvek dva body v polygone $\mathcal{P} \oplus (-\mathcal{R}(x, y))$ je tvorená z úsečiek patriacim konvexným polygonom $\mathcal{P}$ a $(-\mathcal{R}(x, y))$, čím je veta dokázaná.

Pripomeňme, že súčet dvoch polygonov je polygon. Ďalej dokážeme, že konvexný obal Minkowského sumy má najviac $n + m$ hrán. Označme si e ako hranu $\mathcal{P} \oplus (-\mathcal{R}(x, y))$ (viď. obr. 2.8). Táto hrana je extrémna v smere normály na túto hranu, čo znamená, že musela vzniknúť sčítaním bodov z $\mathcal{P}$ a $(-\mathcal{R}(x, y))$, ktoré sú extrémne v tomto smere. Naviac aspoň jeden z polygonov musí mať extrémnu hranu v tomto smere. V prípade zobrazenom na obr. 2.8 je táto hrana označená ako e'. Takýmto spôsobom môžeme označiť každú hranu najviac raz, teda celkový počet hrán je najviac $n+m$. Poznamenajme, že, ak $\mathcal{P}$ a $(-\mathcal{R}(x, y))$ nemajú žiadne dve hrany rovnobežné, tak počet hrán $\mathcal{CP}$ je práve $n + m$.



Obr. 2.8: počet hrán $\mathcal{CP}$ [1]

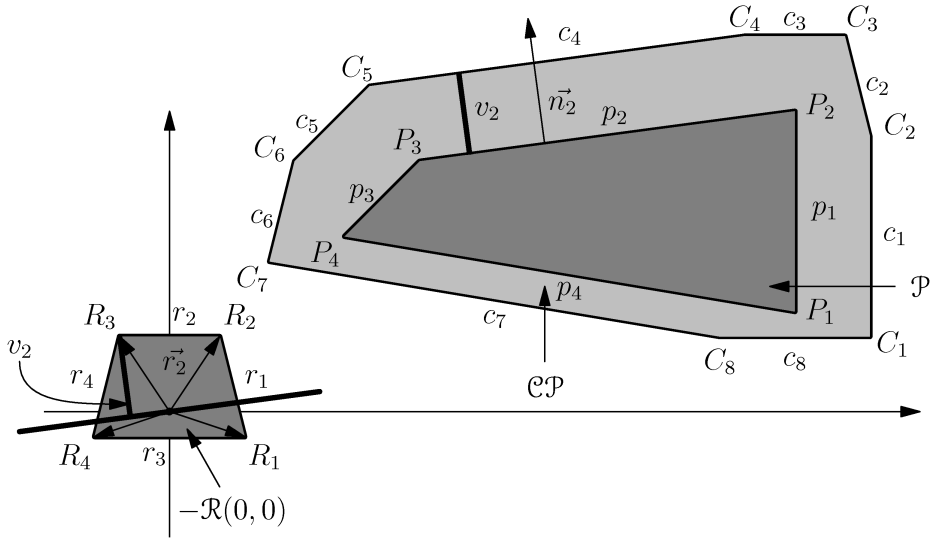
□

Veta 2.5. *Majme robota $-\mathcal{R}(x, y)$, prekážku $\mathcal{P}$ a množinu $\mathcal{CP} = \mathcal{P} \oplus (-\mathcal{R}(x, y))$. Taktiež uvažujme bod $O = (o_x, o_y) \in \mathcal{R}(x, y)$. Obsah $S(\mathcal{CP})$ prekážky v konfiguračnom priestore $\mathcal{CP}$ sa vypočíta ako:*

$$S(\mathcal{CP}) = S(\mathcal{R}(x, y)) + S(\mathcal{P}) + \sum_{i=1}^{n} |p_i||v_i|,$$

kde $S(\mathcal{R}(x, y))$ a $S(\mathcal{P})$ sú obsahy polygonov $\mathcal{R}(x, y)$ a $\mathcal{P}$; p_i je hrana prekážky $\mathcal{P}$; v_i je kolmá vzdialenosť medzi extrémnym bodom robota $\mathcal{R}(x, y)$ v smere n_i a priamkou, ktorá je rovnobežná s hranou p_i a prechádza bodom $O = (o_x, o_y)$, kde n_i je vonkajší normálový vektor hrany p_i.

Dôkaz. Túto vetu dokážeme bez újmy na všeobecnosti pre prípad, že bod O označuje počiatok súradnicovej sústavy, je súčasťou množiny robota a zároveň je aj referenčným bodom robota. Teda platí: $O = (0, 0) \in -\mathcal{R}(0, 0)$. Ďalej si označme vrcholy prekážky ako $P_1, P_2, \ldots, P_n$ a vrcholy robota ako $R_1, R_2, \ldots, R_m$. Taktiež polohové vektory vrcholov robota označíme ako $\vec{r_1}, \vec{r_2}, \ldots, \vec{r_m}$. Jednotlivé hrany prekážky pre $i = 1, \ldots, n$ označíme ako $p_i = P_i P_{i+1}$, pričom $P_{n+1} = P_1$. Podobne označíme aj jednotlivé hrany robota pre $j = 1, \ldots, m$ ako $r_j = R_j R_{j+1}$, pričom $R_{m+1} = R_1$.



Obr. 2.9: príklad ozančenia polygonov

Pripomeňme, že ak robot a prekážka nemajú žiadne rovnobežné hrany, tak počet hrán Minkowského sumy robota a prekážky je práve $m + n$. V prípade, že existujú dve rovnobežné hrany, tak Minkowského suma bude obsahovať hranu, ktorá je taktiež rovnobežná s týmito hranami a jej dĺžka bude rovná súčtu dĺžok jednotlivých hrán. Teda na tejto hrane dokážeme nájsť bod, ktorý bude túto úsečku deliť na dve, ktorých dĺžky budú zodpovedať dĺžkam rovnobežných hrán robota a prekážky.

Označme vrcholy $\mathcal{CP}$ ako $C_1, C_2, \ldots, C_{m+n}$. Keďže každá hrana $\mathcal{CP}$ je rovonobežná s niektorou z hrán robota alebo prekážky, tak môžeme hrany $\mathcal{CP}$ označiť ako $c_k^i = C_k C_{k+1}$, alebo $c_k^j = C_k C_{k+1}$, pričom horný index označuje, či sa jedná o hranu rovnobežnú s robotom (index i), alebo s prekážkou (index j). Číslo i alebo j označuje poradie, s ktorou hranou z robota, alebo prekážky je daná hrana $\mathcal{CP}$ rovnobežná. Pre každé k existuje práve jeden index i, alebo j a pre každé i a j existuje práve jedno k (napríklad hrana $c_k^i = C_k C_{k+1}$ je rovnobežná s hranou p_i z prekážky $\mathcal{P}$). Poznamenajme, že toto indexovanie v praxi nemá význam, ide iba o abstrakciu, ktorá bude ďalej v dôkaze využívaná.

Obsah $\mathcal{CP}$ môžeme rozdeliť na dve časti. Prvú časť tvorí obsah prekážky $\mathcal{P}$, ktorá leží celá v $\mathcal{CP}$, pretože predpokládame, že bod $(0,0) \in \mathcal{R}(0,0)$. Druhú časť tvoria trojuholníky a rovnobežníky, ktoré dopĺňajú zvyšnú časť obsahu $\mathcal{CP}$ nasledovne:

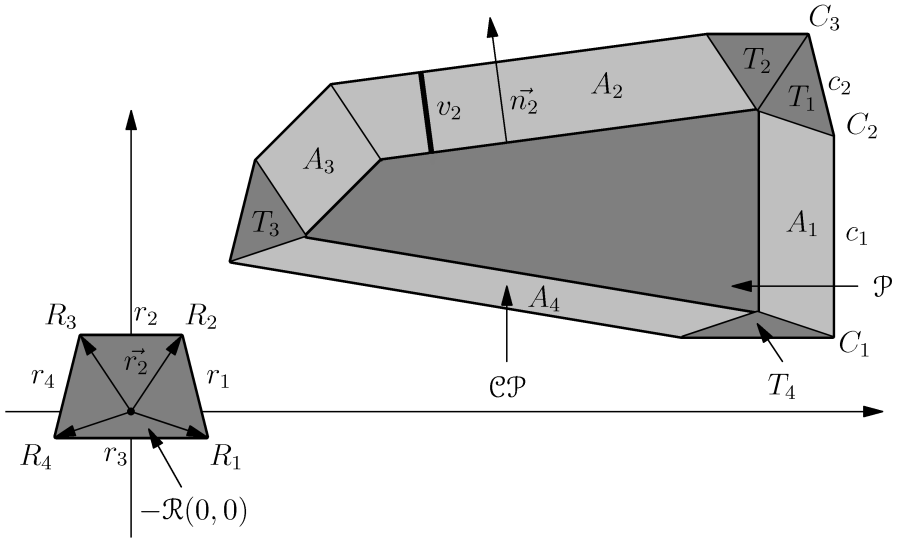
1. pre každú hranu a_i existuje hrana c_k^i pre ktoré platí, že sú rovnobežné a majú rovnakú dĺžku (teda platí, že $p_i || c_k^i$ a $|p_i| = |c_k^i|$). Hranu c_k^i získame tak, že hranu p_i posunieme v smere polohového vektoru $\vec{r}_j$ bodu R_j, ktorý je extrémny v smere normály n_i na hranu p_i. Teda platí, že $C_k = P_i + \vec{r}_j$. Teraz môžeme vytvoriť rovnobežník A_i pre každú jednu hranu p_i, a to tak, že dve hrany tohto rovnobežníka sú $p_i = P_i P_{i+1}$ a $c_i^k = C_k C_{k+1}$, pričom $p_i || c_i^k$. Zvyšné dve hrany sú ekvivalentné a rovnobežné s polohovým vektorom $\vec{r}_j$ bodu R_j, ktorý je extrémny v smere normály n_i na hranu p_i. Obsah rovnobežníka A_i sa vypočíta nasledovne:

   $$S(A_i) = |p_i||v_i|,$$

   kde v_i je výška rovnobežníka A_i na hranu p_i. Veľkosť v_i je rovnaká ako kolmá vzdialenosť medzi extrémnym bodom robota $\mathcal{R}(0,0)$ v smere n_i a priamkou, ktorá je rovnobežná s hranou p_i a prechádza bodom $(0,0)$.

2. hrany $c_k^j = C_k C_{k+1}$ sú rovnobežné s hranami robota. Pretože platí, že $C_k = P_i + \vec{r_j}$ a $C_{k+1} = P_i + \vec{r_{j+1}}$, tak môžeme vytvoriť trojuholník $T_j = C_k C_{k+1} P_i$ pre každú z hrán c_k^j. Tento trojuholník je ekvivalentný s trojuholníkom v OR_jR_{j+1}. Suma týchto trojuholníkov nám dáva plochu robota $\mathcal{R}(0,0)$ a teda platí:

$$\sum_{j=1}^{m} S(T_j) = S(\mathcal{R}(0,0)).$$



Obr. 2.10: príklad rozkladu $\mathcal{CP}$

Prekážka $\mathcal{P}$, rovnobežníky A_i a trojuholníky T_j vypĺňajú celý obsah prekážky v konfiguračnom priestore $\mathcal{CP}$, pretože jediné možné situácie ktoré môžu nastať, sú tieto tri:

1. obe hrany c_k^i a c_{k+1}^{i+1} sú rovnobežné s hranami prekážky $\mathcal{P}$ a rovnobežníky A_i a A_{i+1} majú spoločnú práve jednu hranu.

2. hrana c_k^i je rovnobežná s hranou prekážky $\mathcal{P}$ a hrana c_{k+1}^j je rovnobežná s hranou robota $\mathcal{R}(0,0)$. V tomto prípade rovnobežník a trojuholník majú jednu spoločnú hranu.

3. obe hrany c_k^j a c_{k+1}^{j+1} sú rovnobežné s hranami robota $\mathcal{R}(0,0)$ a trojuholníky T_j a T_{j+1} majú spoločnú práve jednu hranu.

Keďže polygon $\mathcal{CP}$ je konvexný, tak trojuholníky a rovnobežníky sa nebudú prekrívať. Obsah prekážky v konfiguračnom priestore $\mathcal{CP}$ môžeme vyjadriť ako súčet obsahu robota $\mathcal{R}(0,0)$, prekážky $\mathcal{P}$ a sumy obsahov rovnobežníkov A_i. Teda platí:

$$S(\mathcal{CP}) = S(\mathcal{R}(0,0)) + S(\mathcal{P}) + \sum_{i=1}^{n} |p_i||v_i|.$$

□

Poznámka 2.2 Keďže platí veta 2.2, tak obsah prekážky v konfiguračnom priestore $\mathcal{CP}$ sa nezmení pre akúkoľvek polohu referenčného bodu (napríklad v prípade, že referenčný bod leží mimo robota).

Kapitola 3

Algoritmus pre výpočet Minkowského sumy

Táto kapitola pojednáva o algoritme počítajúcom Minkowského sumu. V úvode tejto kapitoly bude uvedený pseudoalgoritmus, ktorý bude na dvoch príkladoch postupne vyladený. Ďalej bude pseudoalgoritmus vyladený a implementovaný do prostredia jazyka C#. Na konci tejto kapitoly bude uvedený stručný popis práce s programom ConfSpace, ktorý je zároveň priložený v prílohe tejto práce. V celej kapitole budú reprezentovať množiny robota $\mathcal{R}(x, y)$ a prekážky $\mathcal{P}$ len vrcholy jednotlivých polygonov.

3.1 Pseudoalgoritmus MinkowskySum

Potrebujeme nájsť algoritmus, ktorý bude efektívne počítať Minkowského sumu, teda zistí tvar prekážok $\mathcal{CP}$. Prirodzene prvé, čo nás napadne, je nasledovný a veľmi jednoduchý algoritmus. Pre každý pár v, w vrcholov, kde $v \in \mathcal{P}$ a $w \in (-\mathcal{R})$, sa spočíta $v+w$. Následne sa vytvorí konvexný obal tejto množiny. Pri prekážkach s veľkým počtom vrcholov, nie je tento algoritmus efektívny, pretože počítanie každého jedného páru v, w je v tomto prípade náročné.

Ako uvádzajú Berg, Cheong a Kreveld [1], existuje algotitmus, ktorý tento problém rieši podstatne efektívnejšie. Namiesto toho, aby sme počítali každý jeden pár v, w, tak využijeme vetu 2.3 a budeme sčítavať iba body, ktoré sú extrémne v niektorých smeroch. Tento algoritmus bude uvedený v pseudokóde, ten bude v ďalšej podkapitole vyladený a nakoniec bude implemetnovaný.

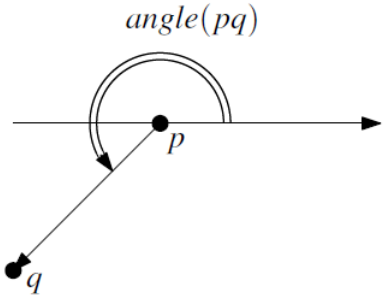
V nasledujúcom pseudokóde sa používa funkcia *angle*(pq), ktorá vracia hodnotu veľkosti uhlu medzi vektorom $\vec{pq}$ a kladným smerom osi x (viď. obr. 3.1).

Teraz môžeme prejsť k pseudokódu algortimu, ktorý nazveme MinkowskySum.

Algoritmus MinkowskySum$(\mathcal{P}, \mathcal{R})$

Vstup: konvexný polygon $\mathcal{P}$ s vrcholmi $p_1, \ldots, p_n$, konvexný polygon $-\mathcal{R}(x, y)$ s vrcholmi $r_1, \ldots, r_n$, kde zoznamy vrcholov sú zoradené proti smeru hodinových ručičiek, kde p_1 a r_1 sú vrcholy s najmenšou y–ovou hodnotou (v prípade rovnakej y–ovej hod–noty dvoch vrcholov s najmenšou x–ovou hodnotou).

Výstup: Mikowského suma $\mathcal{P} \oplus (-\mathcal{R}(x, y))$.



Obr. 3.1: funkcia $angle(pq)$

$i \leftarrow 1$; $j \leftarrow 1$
$p_{n+1} \leftarrow p_1$; $p_{n+2} \leftarrow p_2$; $r_{n+1} \leftarrow r_1$; $r_{n+2} \leftarrow r_2$
repeat
pridaj $p_i + r_j$ ako vrchol $\mathcal{P} \oplus (-\mathcal{R}(x, y))$
if $angle(p_i p_{i+1}) < angle(r_j r_{j+1})$
then $i \leftarrow i + 1$
else if $angle(p_i p_{i+1}) > angle(r_j r_{j+1})$
then $j \leftarrow j + 1$
else $i \leftarrow i + 1$; $j \leftarrow j + 1$
until $i = n + 1$ **and** $j = m + 1$

Tento pseudoalgoritmus, ktorý uvádzajú vo svojej publikácii Berg, Cheong a Kreveld [1], nefunguje správne pre všetky prípady. Na záver tejto podkapitoly uvedieme, že časová náročnosť výpočtu Minkowského sumy dvoch polygonov, ktoré majú m a n vrcholov, pomocou algoritmu MinkowskySum je $O(n + m)$.

3.2 Ladenie algoritmu MinkowskySum

V tejto podkapitole budú uvedené dva príklady, na ktorých sa vyladí algoritmus MinkowskySum. V prvom príklade, s rovnakým zadaním ako v príklade 2.1, objasníme fungovanie algoritmu a poukážeme na prvú chybu. Týmto algoritmus mierne odladíme. Pridáme ešte jeden príklad, v ktorom demonštrujeme, že ani v tomto prípade algoritmus nefunguje pre všetky prípady, pričom vyvodíme ďalšiu miernu úpravu, aby fungoval pre všetky prípady.

Príklad 3.1
Majme prekážku a robota zadané nasledovne:

$$\mathcal{P} = \{(40, 40), (80, 40), (80, 80), (40, 80)\}$$
$$\mathcal{R}(0, 0) = \{(0, 0), (0, 20), (20, 0)\}.$$

Teda vstup algoritmu bude tvorený prekážkou $\mathcal{P}$ a zaporným robotom $-\mathcal{R}(0, 0)$. Robot je záporný, pretože chceme vypočítať tvar prekážky $\mathcal{CP}$, a teda platí veta 2.1. Záporný robot:

$$-\mathcal{R}(0, 0)) = \{(0, -20), (0, 0), (-20, 0)\}.$$

Ako vidíme, všetky podmienky zo vstupu sú splnené. Obe množiny majú prvý bod s najmenšou y–ovu hodnotou, pričom jednotlivé body sú zoradené proti smeru hodinových ručičiek. Môžeme pristúpiť k postupnému aplikovaniu algoritmu.

Vytvoríme si premenné i a j, ktoré zapíšeme do tabuľky a postupne budeme zaznamenávať ich zmeny počas cyklov (viď. tab. 3.2). Taktiež k množinám $\mathcal{P}$ a $-\mathcal{R}(0,0)$ pridáme na konce zonznamov po dva body, s rovnakými hodnotami, ako majú prvé dva body v týchto množinách. V celom algoritme budeme potrebovať uhly, ktoré zvierajú polygon s kladnou osou x. Vytvoríme si ďalšie dve tabuľky. Do prvej zapíšeme jednotlivé, už rozšírené množiny, a taktiež uhly (viď. Tab. 3.1). Do druhej zapíšeme body získané každým cyklom, teda vrcholy $\mathcal{CP}$ (viď. Tab. 3.3).

	$0°$	$90°$	$180°$	$270°$	$0°$	
i	1	2	3	4	5	6
$\mathcal{P}$	$(40,40)$	$(80,40)$	$(80,80)$	$(40,80)$	$(40,40)$	$(80,40)$
	$90°$	$180°$	$315°$	$90°$		
j	1	2	3	4	5	
$-\mathcal{R}(0,0)$	$(0,-20)$	$(0,0)$	$(-20,0)$	$(0,-20)$	$(0,0)$	

Tabuľka 3.1: vstupné hodnoty

Na začiatku prvého cyklu okamžite získavame prvý bod Minkowského sumy. Sčítame prvé body z každej množiny, čím dostávame bod $(40,20)$. Porovnáme uhly a zväčšíme premennú i.

Pokračujeme druhým cyklom, kde sčítame druhý bod prekážky s prvým bodom robota. Tento bod má hodnotu $(80,20)$. Po porovnaní uhlov v tomto prípade zväčšíme obe premenné i a j.

V tomto cykle po sčítaní tretieho bodu prekážky s druhým bodom robota dostávame bod $(80,80)$. Prejdeme k porovnaniu uhlov a opäť zväčšíme obe premenné i a j.

Cyklus ešte stále nespĺňa podmienku ukončenia a teda prechádzame k ďalšiemu sčítaniu bodov, kedy získame bod $(20,80)$. V tomto cykle po porovnaní uhlov zväčšujeme premennú i.

Teraz je už podmienka na ukončenie cyklu čiastočne splnená, čo ale nestačí. Po sčítaní určených bodov dostávame bod $(20,40)$. Týmto bodom dostaváme kompletnú množinu bodov $\mathcal{CP}$, pretože je identická s výslednou množinou v príklade 2.1. Napriek tomu algoritmus nekončí, pretože ak porovnáme uhly tak zisťujeme, že opäť zväčšujeme premennú i a tým pádom už nikdy nedosiahneme podmienku na ukončenie cyklu a cyklus pretečie.

číslo cyklu	1	2	3	4	5
i	2	3	4	5	6
j	1	2	3	3	3

Tabuľka 3.2: zmeny premenných počas cyklov

Ako riešenie sa ponúka upraviť podmienku na ukončenie cyklu. Ak by sme v tejto podmienke vymenili logickú spojku **and** za **or**, tak to stále nie je ideálne riešenie, pretože cyklus by sa ukončil ešte skôr, ako by sme dostali posledný bod. Aby bol ošetrný aj tento problém, sčítanie bodov z množín bude prebiehať až po zväčšovaní premenných i

číslo cyklu	1	2	3	4	5
$\mathcal{CP}$	$(40,20)$	$(80,20)$	$(80,80)$	$(20,80)$	$(20,40)$

Tabuľka 3.3: vrcholy $\mathcal{CP}$

a j. Prvé sčítanie prebehne ešte pred začiatkom cyklu. Takto upravený pseudoalgoritmus vyzerá nasledovne:

$i \leftarrow 1$; $j \leftarrow 1$
$p_{n+1} \leftarrow p_1$; $p_{n+2} \leftarrow p_2$; $r_{n+1} \leftarrow r_1$; $r_{n+2} \leftarrow r_2$
pridaj $p_1 + r_1$ ako vrchol $\mathcal{P} \oplus (-\mathcal{R}(x,y))$
repeat
    if $angle(p_i p_{i+1}) < angle(r_j r_{j+1})$
    then $i \leftarrow i + 1$
    else if $angle(p_i p_{i+1}) > angle(r_j r_{j+1})$
        then $j \leftarrow j + 1$
        else $i \leftarrow i + 1$; $j \leftarrow j + 1$
    pridaj $p_i + r_j$ ako vrchol $\mathcal{P} \oplus (-\mathcal{R}(x,y))$
until $i = n + 1$ **or** $j = m + 1$

Tento algoritmus aplikujeme v nasledujúcom príklade.

Príklad 3.2
Majme prekážku a robota zadané nasledovne:

$$\mathcal{P} = \{(40,40),(80,40),(80,80),(40,80)\}$$
$$\mathcal{R}(0,0) = \{(0,0),(0,20),(15,15),(20,0)\}.$$

Robot bude prevedený na záporného a bude upravené poradie bodov, čím dostávame túto množinu:

$$\mathcal{R}(0,0) = \{(0,-20),(0,0),(-20,0),(-15,-15)\}.$$

Rovnako ako v predchádzajúcom príklade, aj v tomto budú vytvorené tri tabuľky (viď. Tab. 3.2, Tab. 3.5, Tab. 3.6).

		0°		90°		180°		270°		0°	
i	1		2		3		4		5		6
$\mathcal{P}$	$(40,40)$		$(80,40)$		$(80,80)$		$(40,80)$		$(40,40)$		$(80,40)$
		90°		180°		288°		341°		90°	
j	1		2		3		4		5		6
$-\mathcal{R}(0,0)$	$(0,-20)$		$(0,0)$		$(-20,0)$		$(-15,-15)$		$(0,-20)$		$(0,0)$

Tabuľka 3.4: vstupné hodnoty

V tomto príklade nebude uvedený podrobný popis algoritmu ako v predchádzajúcom príklade, pretože algoritmus je len mierne pozmenený. V tomto prípade algoritmus nepretečie a ako výsledok dostávame množinu bodov. Napriek zmenám v algoritme, ale aj

číslo cyklu	1	2	3	4
i	2	3	4	5
j	1	2	3	3

Tabuľka 3.5: zmeny premenných počas cyklov

číslo cyklu	1	2	3	4	5
$\mathcal{CP}$	$(40, 20)$	$(80, 20)$	$(80, 80)$	$(20, 80)$	$(20, 40)$

Tabuľka 3.6: vrcholy $\mathcal{CP}$

v zadaní robota, výsledkom tohto algoritmu je rovnaká množina bodov $\mathcal{CP}$ ako v predchádzajúcom príklade. Už je možné vytušiť, že výsledok nie je správny. Ak sa prevedie analytický vypočet, ako v príklade 2.1 pre toto zadanie, tak zistíme, že výsledkom je takáto množina bodov:

$$\{(40, 20), (80, 20), (80, 80), (20, 80), (20, 40), (25, 25)\},$$

čo dokazuje, že jeden bod chýba.

Problém v tomto prípade nastáva pri porovnávaní uhlov medzi 3. a 4. bodom v množine $-\mathcal{R}(0, 0)$ s uhlami z množiny $\mathcal{P}$. Tetno uhol je väčší ako všetky uhly, s ktorými bude porovnávaný, a teda nikdy nenastane situácia, aby bola zväčšená premenná j, čo by spôsobilo posun. Riešením tejto situácie je pričítanie 360° k uhlom, ktoré vytvárajú práve tie dva pridané body. Pri tejto úprave je ešte potrebné opäť zmeniť logickú spojku v podmienke na ukončenie cyklu z **or** späť na **and**. Takto upravný algoritmus MinkowskySum môžeme v pseudokóde zapísať takto:

$i \leftarrow 1$; $j \leftarrow 1$
$p_{n+1} \leftarrow p_1$; $p_{n+2} \leftarrow p_2$; $r_{n+1} \leftarrow r_1$; $r_{n+2} \leftarrow r_2$
pridaj $p_1 + r_1$ ako vrchol $\mathcal{P} \oplus (-\mathcal{R}(x, y))$
repeat
    if $i == n - 1$
    then $angle(p_i p_{i+1}) \leftarrow angle(p_i p_{i+1}) + 360°$
    if $j == n - 1$
    then $angle(r_j r_{j+1}) \leftarrow angle(r_j r_{j+1}) + 360°$

    if $angle(p_i p_{i+1}) < angle(r_j r_{j+1})$
    then $i \leftarrow i + 1$
    else if $angle(p_i p_{i+1}) > angle(r_j r_{j+1})$
        then $j \leftarrow j + 1$
        else $i \leftarrow i + 1$; $j \leftarrow j + 1$
    pridaj $p_i + r_j$ ako vrchol $\mathcal{P} \oplus (-\mathcal{R}(x, y))$
until $i = n + 1$ **or** $j = m + 1$

V nasledujúcej podkapitole bude algoritmus implementovaný.

Poznámka 3.1 Ak tento algoritmus aplikujeme na každú prekážku $\mathcal{P}$ z množiny prekážok S, tak získame množinu prekážok v konfiguračnom priestore, čo nie je nič iné ako zakázaný konfiguračný priestor, teda $\mathcal{C}_{forb}(\mathcal{R}, S)$.

Poznámka 3.2 Tento upravený algoritmus sa nazýva MinkowskySum a pri odvolávaní sa naň v ďalšom texte budeme stále uvažovať vyžšie uvedený pseudoalgoritmus z príkladu 3.2. Vstup algoritmu sa pri každej modifikácií uvažuje rovnaký, pričom správny výstup algoritmu je dosahnutý až po úprave z posledného príkladu.

Poznámka 3.3 V programe priloženom v prílohe príklady 3.1 a 3.2 sú implementované aj s vyžšie popisovanými chybami. Kompletné zadania sa zobrazia po kliknutí na tlačitko Príklad 3.1 alebo Príklad 3.2.

3.3 Implementácia algoritmu

Algoritmus MinkowskySum implementujeme v prostredí jazyka C#. Budeme sa musieť vysporiadať s niekoľkými problémami a budeme musieť vytvoriť niekoľko pomocných metód. Pre implementáciu algoritmu MinkowskySum vytvoríme novú triedu, ktorú pomenujeme `MinkowskySum`.

Ako prvé, bude vytvorená metóda, v ktorej budeme počítať uhol medzi dvoma vektormi. Ukážka tejto metódy je v zdrojovom kóde 1. Argumenty tejto metódy budú dva body, pomocou ktorých vytvoríme vektor $\vec{u}$, ako je vidieť na riadkoch 7. a 8. Druhý vektor bude jednotkový vektor v smere osi x, a teda $\vec{v} = (1, 0)$, ktorý je vytvorený na riadkoch 5. a 6. Pre výpočet uhlu medzi týmito dvoma vektormi využijeme známy vzorec:

$$\varphi = \arccos\left(\frac{u_x v_x + u_y v_y}{|\vec{u}| \cdot |\vec{v}|}\right); \qquad 0° \leq \varphi \leq 180°.$$

V ukážke kódu, ktorý je uvedený nižšie, je tento vzorec implementovaný na riadkoch 9, 10 a 11. Ďalej je vzorec trošku modifikovaný, ako je vidieť na riadku 12, pretože funkcia `Math.Acos` nám C# vracia hodnotu v radiánoch. Táto modifikácia je vytváraná len z dôvodu lepšej predstavy o uhle pri ladení algoritmu. Ako si môžeme všimnúť, vzorec pre výpočet uhlu nám nadobúda hodnoty iba od 0° do 180°. Túto hodnotu prekročíme jedine v prípade, že y zložka vektoru $\vec{u}$ je záporná. V takomto prípade odčítame od plného uhla 360° získaný uhol `tmpUhol`. Výsledkom je uhol medzi vektorom určeným dvoma zadanými bodmi a kladným smerom osi x v stupňoch.

Ďalším problémom, ktorý je potrebné vyriešiť, je zabezpečiť, že podmienky pri vstupe v algoritme MinkowskySum určite nastanú. Konkrétne potrebujeme zaručiť, aby jednotlivé zoznamy vrcholov každého z polygonov boli zoradené proti smeru hodinových ručičiek a súčasne, aby prvý bod zoznamu bol ten s najmenšou y–ovou hodnotou, v prípade rovnosti, ten s najmešou x–ovou hodnotou. Podmienku, aby zoznam bol usporiadaný v správnom smere, teraz implementujeme.

Ešte predtým ako sa pustíme do implementácie, si musíme objasniť, že v danom programe je každý polygon reprezentovaný zoznamom bodov, teda vrcholov polygonu. Všetky vrcholy sú typu `Bod`. Tetno typ bolo potrebné vytvoriť a jeho bližší popis si uvedieme neskôr. Jednotlivé vrcholy sú uložené v rozhraní `List<>`.

Nasledujúcim problémom je potrebné zistiť, či jednotlivé body v zozname sú usporiadané proti, alebo v smere hodinových ručičiek. Tento problém vyrieši vektorový súčin.

```
 1.private double AngleMS(Bod P, Bod Q)
 2.        {
 3.            Bod v = new Bod();
 4.            Bod u = new Bod();
 5.            v.X = 1;
 6.            v.Y = 0;
 7.            u.X = Q.X - P.X;
 8.            u.Y = Q.Y - P.Y;
 9.            double tmpUhol = Math.Acos((v.X * u.X + v.Y * u.Y) /
10.                (Math.Sqrt(Math.Pow(u.X, 2) + Math.Pow(u.Y, 2)) *
11.                Math.Sqrt(Math.Pow(v.X, 2) + Math.Pow(v.Y, 2))));
12.            tmpUhol = tmpUhol * 180 / Math.PI;
13.            if (u.Y < 0)
14.                tmpUhol = 360 - tmpUhol;
15.            return tmpUhol;
16.        }
```

Zdrojový kód 1: počítanie uhla medzi vektorom $\vec{pq}$ a kladným smerom osi x

Najskôr vytvoríme metódu `DirectionOrder`, ktorá bude vyžadovať v argumente zoznam bodov polygonu.

Ako už bolo spomenuté, vektorový súčin bude nápomocný pri zisťovaní orientácie poradia bodov. Na jednoduchom príklade bude ukázané, ako postupovať. Majme trojuholník v rovine xy, čo je polygon, ktorý má tri vrcholy, pričom tieto vrcholy budú zapísané v zozname a bude potrebné vyriešiť rovnaký problém s orientáciu. Označme si vrcholy ako A, B a C. Postupne vytvoríme dva vektory a to tak, že budeme brať za sebou body zo zoznamu, a teda vzniknú nám vektory $\vec{u} = B - A$ a $\vec{v} = C - B$. Teraz pridáme tretí rozmer. Jednotlivé body aj vektory budú mať z–ovú súradnicu nulovú. Ak vektorovo vynásobíme vektory $\vec{u}$ a $\vec{v}$, tak nám vzniká tretí vektor $\vec{w}$, ktorý je rovnobežný s osou z a kolmý na trojuholník. O tom, aký smer bude mať vektor $\vec{w}$, či bude smerovať "hore", alebo "dole", rozhoduje jeho tretia zložka. Podľa definíce

$$\vec{w} = \vec{u} \times \vec{v} = (u_2v_3 - v_2u_3; u_3v_1 - v_3u_1; u_1v_2 - v_1u_2)$$

má tretia zložka tvar $u_1v_2 - v_1u_2$. Týmto sa dozvieme smer, a teda zistíme ako sú orientované body v zozname. Ak je posledná zložka vektoru $\vec{w}$ záporná, tak body sú zapísané v zozname v smere hodinových ručičiek a je potrebné zoznam prepísať. V opačnom prípade sú body zapísané správne. Tento postup sa nazýva aj ako "pravidlo pravej ruky".

Ukážka tejto metódy je zobrazená v zdrojovom kóde 2. Na riadkoch 3., 4., 5. a 6. sú vytvorené vektory $\vec{u}$ a $\vec{v}$ z prvých troch bodov v zozname. Na riadkoch 7. a 8. sa symbolicky rozširuje priestor z dvojdimenzionálneho na trojdimenzionálny. Kľúčový je riadok 10., kde sa počíta tretia zložka vektorového súčinu. Ďalej je zavedená podmienka, ktorá tým najjednoduchším spôsobom poprehadzuje zoznam bodov v prípade, že orientácia je nevyhovujúca. Tento zdrojový kód obsahuje časti kódu, ktoré nie sú pre chod programu nevyhnutné. Sú tam uvedené pre lepšiu ilustráciu riešenia daného problému.

```
 1. public void DirectionOrder(List<Bod> ListUpravit)
 2. {
 3.             Bod u = new Bod(ListUpravit[1].X - ListUpravit[0].X,
 4.                 ListUpravit[1].Y - ListUpravit[0].Y);
 5.             Bod v = new Bod(ListUpravit[2].X - ListUpravit[1].X,
 6.                 ListUpravit[2].Y - ListUpravit[1].Y);
 7.             int[] vektorU = { u.X, u.Y, 0 };
 8.             int[] vektorV = { v.X, v.Y, 0 };
 9.             int vektorUxV;
10.             vektorUxV = vektorU[0]*vektorV[1]-vektorU[1]*vektorV[0];

11.             if (vektorUxV < 0)
12.             {
13.                 List<Bod> tmpList = new List<Bod>();
14.                 for (int i = 0; i < ListUpravit.Count; i++)
15.                 {
16.                     tmpList.Add(ListUpravit[i]);
17.                 }
18.                 for (int i = 0; i < ListUpravit.Count; i++)
19.                 {
20.                     ListUpravit[i] = tmpList[ListUpravit.Count-i-1];
21.                 }
22.             }
23.         }
```

Zdrojový kód 2: zisťovanie smeru a úprava zoznamu bodov

Je potrebné ešte zabezpečiť druhú podmienku vo vstupe algoritmu MINKOWSKY-SUM, ktorá výžaduje, aby prvý bod v zozname bodov bol ten s najmenšou y–ovou hodnotou a v prípade rovnosti s najmenšou x–ovou hodnotou. Vytvoríme ešte jednu metódu, ktorá nájde tento bod a patrične upraví zoznam. Implementovať túto metódu nie je náročné a preto ukážku zdrojového kódu vynecháme. V programe je táto metóda implementovaná ako `CorrectOrder`.

Konečne je všetko pripravené k tomu, aby mohla byť vytvorená metóda, ktorá vypočíta Minkowského sumu a volá sa `Pocitaj`. Konkrétna ukážka implementácie je k nahliadnutiu nižšie (zdrojový kód 3). Rozdiel od pseudokódu je na riadkoch 4. až 8. kde sa prevádza robot na záporného. Riadky 9., 10., 11. a 12. sú venované úprave jednotlivých zoznamov bodov tak, aby boli splnené vstupné podmienky. Premenné i a j sú vytvorené na riadkoch 13. a 14. Ďalej sa deklarujú premenné `UholR` a `UholP`, do ktorých sa neskôr budú ukladať vypočítané uhly a následne navzájom porovnávať. Na riadkoch 17. až 24. sú na konce zoznamov pridávané body s rovnakými hodnotami, ako majú body na začiatku zoznamov. Pred začiatkom cyklu na riadkoch 25. až 28. sa spočíta prvý bod a uloží sa do zoznamu, ktorý zároveň tvorí výstup celej metódy. V pseudokóde je použitý cyklus `repeat - while`, ktorý v prostredí C# nie je dostupný. Ako alternatíva sa ponúka cyklus `while`. Aby nebolo potrebné počítať uhol pri každom `if`, tak sa spočíta uhol stále na začiatku každého cyklu a uloží sa do už spomínaných premenných, ako je uvedené na riadkoch 30. a 31. Nasledujúce riadky sú už takmer identické s pseudokódom. Na záver, na riadku 44., sa vracia zoznam vypočítaných vrcholov, čo sú vrcholy prekážky v konfiguračnom priestore $\mathcal{CP}$.

```
 1.public List<Bod> Pocitaj(List<Bod> Robot, List<Bod> Prekazka)
 2.       {
 3.           List<Bod> tmpR = new List<Bod>();
 4.           for (int k = 0; k < Robot.Count; k++) {
 5.               int a = -Robot[k].X;
 6.               int b = -Robot[k].Y;
 7.               Bod c = new Bod(a, b);
 8.               tmpR.Add(c); }
 9.            DirectionOrder(tmpR);
10.            CorrectOder(tmpR);
11.            DirectionOrder(Prekazka);
12.            CorrectOder(Prekazka);
13.            int i = 0;
14.            int j = 0;
15.            double UholR = 0;
16.            double UholP = 0;
17.            Bod vn1 = new Bod(tmpR[0].X, tmpR[0].Y);
18.            Bod vn2 = new Bod(tmpR[1].X, tmpR[1].Y);
19.            Bod wn1 = new Bod(Prekazka[0].X, Prekazka[0].Y);
20.            Bod wn2 = new Bod(Prekazka[1].X, Prekazka[1].Y);
21.            tmpR.Add(vn1);
22.            tmpR.Add(vn2);
23.            Prekazka.Add(wn1);
24.            Prekazka.Add(wn2);
25.            Bod tmpPoint1 = new Bod();
26.            tmpPoint1.X = tmpR[0].X + Prekazka[0].X;
27.            tmpPoint1.Y = tmpR[0].Y + Prekazka[0].Y;
28.            Vrcholy.Add(tmpPoint1);

29.            while ((i<=(Prekazka.Count - 2))&&(j<=(tmpR.Count - 2))) {
30.                UholP = AngleMS(Prekazka[i], Prekazka[i + 1]);
31.                UholR = AngleMS(tmpR[j], tmpR[j + 1]);
32.                if (i == (Prekazka.Count - 2))
33.                    UholP = UholP + 361;
34.                if (j == (tmpR.Count - 2))
35.                    UholR = UholR + 361;
36.                if (UholR > UholP) { i++; }
37.                else
38.                    if (UholR < UholP) { j++; }
39.                    else { i++; j++; }
40.                Bod tmpPoint = new Bod();
41.                tmpPoint.X = tmpR[j].X + Prekazka[i].X;
42.                tmpPoint.Y = tmpR[j].Y + Prekazka[i].Y;
43.                Vrcholy.Add(tmpPoint); }
44.            return Vrcholy;
45.       }
```

Zdrojový kód 3: implementovaný pseudokód MinkowskySum

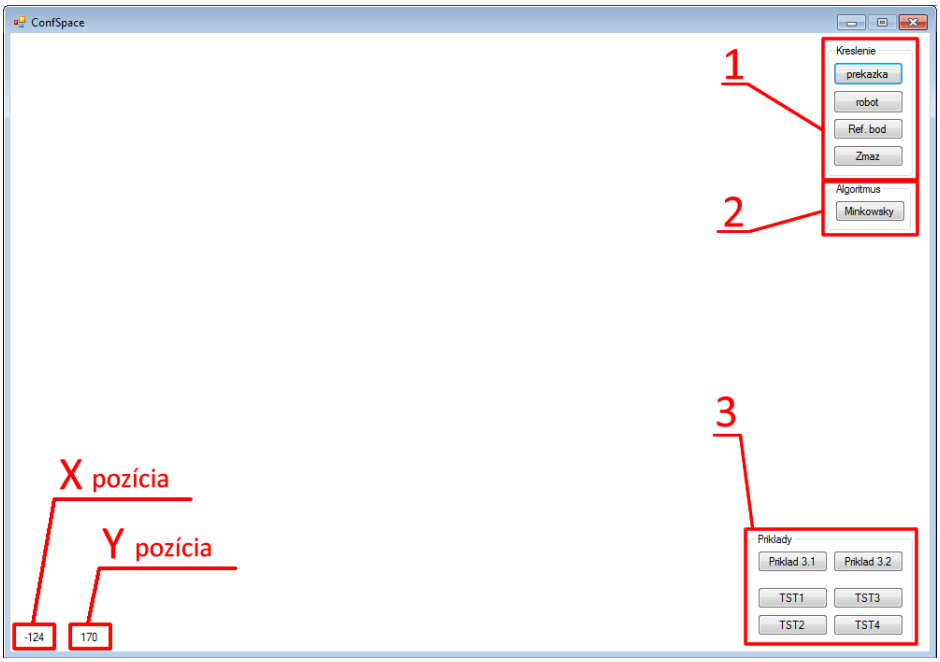
Poznámka 3.4 Ak sa v pracovnom priestore nachádza viacero prekážok, je potrebné vytvoriť cyklus, v ktorom sa bude volať metóda `Počítaj` pre každú jednu prekážku.

Vráťme sa ešte k dátovému typu `Bod`. Je potrebné, aby tento dátový typ bol vytvorený. Dátový typ `Bod` prijmá dva argumenty, ktoré udávajú jeho polohu a teda má dve vlastnosti `X` a `Y`, čo sú súradnice charakterizujúce daný bod. Ak sa niekde v kóde vytvára instancia triedy `Bod`, tak je možné nezadávať argumenty, čím bude vytvorený nový bod so súradnicami $(0, 0)$. V opačnom prípade, ak budú súradnice zadané, tak bude vytvorený bod na danej pozícii. Taktiež je veľmi dôležité implementovať rozhranie `IComperable<>`, ktoré sa využíva v metóde `CorrectOrder`. Pri porovnávaní bodov, teda pri zisťovaní, ktorý bod má najmenšiu y–ovú hodnotu, a v prípade zhody, najmenšiu x–ovú hodnotu.

Na záver tejto podkapitoly je potrebné poznamenať, že sú tu uvedené len tie najdôležitejšie metódy. Pre hladký chod programu, je potrebné urobiť ešte niekoľko doplnení mimo týchto metód a tejto triedy. Tieto doplnenia je možné vidieť v priloženom programe ConfSpace.

3.4 Užívateľský manuál programu ConfSpace

Program ConfSpace počíta Minkowského sumu robota a prekážok. Tento program je priložený v prílohe a v tejto podkapitole bude uvedený krátky užívateľský manuál. Po spustení programu `ConfSpace.exe` sa zobrazí okno, ktoré obsahuje tri kontajnery s rôznymi tlačidlami a indikátormi pozície kurzoru. Celá zvyšná plocha je určená ku kresleniu prekážok a robota a je to teda pracovný priestor robota, ktorý bude postupne zapĺňaný. Označme jednotlivé kontajnery číslami 1, 2 a 3 (viď. obr. 3.2). Spôsob práce s programom a funkcia tlačidiel v kontajneroch 1 a 2 budú vysvetlené súčasne.



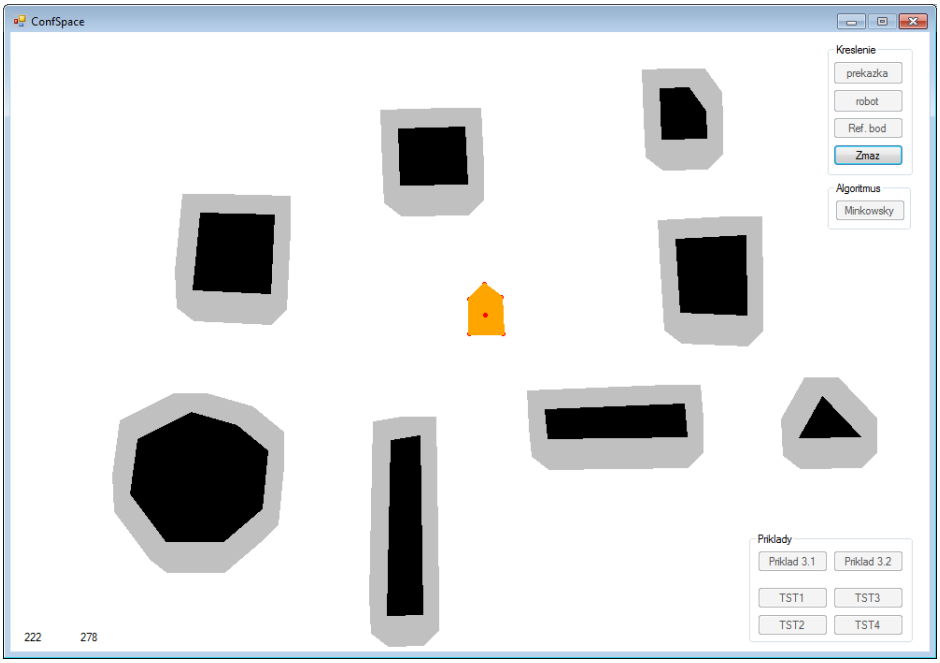
Obr. 3.2: označenie kontajnerov

Kontajner číslo 1 sa nazýva Kreslenie. Obsahuje štyri tlačidlá. Prvé tlačidlo sa nazýva prekážka. Skôr ako bude toto tlačidlo stlačené, je potrebné zadať vrcholy prekážky na pra-

covný priestor. Zadávanie vrcholov prebieha kliknutím na voľný pracovný priestor. Kliknutím vznikne červená bodka, ktorá ozančuje vrchol polygonu. Program je stavaný tak, že každý nový bod bude tvoriť s tým predchádzajúcim hranu polygonu. Preto je dôležité dávať pozor aby nedošlo k pretnutiu hrán. Taktiež body, ktoré budú zadané v pracovnom priestore musia tvoriť konvexný polygon. Ak sú vrcholy prekážky zadané na pracovnom priestore, tak sa po stalačení tlačidla prekazka zobrazí čierna prekážka na pracovej ploche. Rovnakým spôsobom je možné zadávať aj ďalšie prekážky.

Pri vytváraní robota je postup veľmi podobný. Opäť sa na pracovný priestor zadajú vrcholy robota a po kliknutí na druhé tlačidlo, s názvom robot, sa z daných vrcholov vytvorí oranžový polygon predstavujúci robota. Týmto sa toto tlačidlo znefunkční, pretože robot môže byť iba jeden. Prichádza na rad určenie referenčného bodu robota. Opäť to prebieha kliknutím na voľný pracovný priestor, alebo na robota. Následne je potrebné kliknuť na tlačídlo Ref. bod, čím sa tento bod uloží ako referenčný bod robota. Týmto sa aj toto tlačidlo znefunkční, pretože robot môže mať iba jeden referenčný bod. V prípade, že užívateľ nešpecifikuje polohu referenčného bodu, tak za referenčný bod sa bude brať počiatok, teda bod $(0, 0)$.

Týmto je na pracovnom priestore určený robot, jeho referenčný bod a prekážky. Ak v kontajneri ozanačenom číslom 2, nazvanom Algoritmus, klikneme na tlačidlo Minkowsky, tak sa zobrazia sivé polygony, ktoré zobrazujú tvar prekážok v konfiguračnom priestore (napríklad viď. obr. 3.3). Na výčistenie pracovného priestoru slúži tlačidlo z kontajneru 1 s názvom Zmaz. Potom je možné zadávať prekážky aj robota znovu.



Obr. 3.3: príklad použitia programu ConfSpace

V kontajneri ozančenom číslom 3 je šesť tlačidiel. Tlačidlá Priklad 3.1 a Priklad 3.2 už boli spomenuté v podkapitole 3.2. Tlačidlá TST1, TST2, TST3 a TST4 slúžia ako zakladné preddefinované príklady tvarov robota a prakážky. Po kliknutí na tieto tlačidlá je ešte stále možné pridávať ďalšie prekážky a taktiež určiť referenčný bod robta.

Záver

Cieľom práce bolo využiť geometrické algoritmy v robotike, konkrétne nájsť voľný konfiguračný priestor pre polygonálneho robota. Hlavným problémom bolo určiť tvar prekážky v konfiguračnom priestore. K tomu, aby bol dosiahnutý daný cieľ, sme definovali Minkowského sumu, ktorá nám pomohla pri výpočte tvaru. Taktiež v práci boli uvedené a dokázané základné vlastnosti Minkowského sumy. Uviedli sme algoritmus MinkowskySum, ktorý vďaka jednej z uvedených vlastnotí počíta Minkowského sumu pomerne efektívne. Na dvoch príkladoch bol tento algoritmus vyladený. V nasledujúcej kapitole bola uvedená jeho implementácia do prostredia C# spolu s niekoľkými pomocnými metódami. Celý program počítajúci tvar prekážok v konfiguračnom priestore a teda voľný konfiguračny priestor je spolu so zdrojovým kódom priložený v prílohe. Týmto sa dá konštatovať, že daný cieľ je splnený.

Literatúra

[1] Berg, M.– Cheong, O.– Kreveld M.: *Computational Geometry: Algorithms and Applications*, 3rd edition, Berlin: Springer, 2008, ISBN: 978-3-540-77973-5.

[2] Choset. H.: *Principles of Robot Motion – Theory, Algorithms and Implementations* , Massachusetts : MIT Press, 2005, ISBN 0-262-03327-5.

[3] O´Rourke, J.: *Computational geometry in C*, 2nd edition, Cambridge : Cambridge University Press, 1998, ISBN 0-521-64976-5

[4] Tomiczková, S.: *Area of the Minkowksy Sum of Two Convex Sets* , 25. Konference o geometrii a počítačové grafice, 2005.

[5] URL: <http://en.wikipedia.org/wiki/Big_Oh_notation> [cit. 2010-03-05]

Zoznam príloh

CD obsahujúce:

1. program ConfSpace.exe
2. projekt ConfSpace (zdrojové kódy programu)
3. text bakalárskej práce vo formáte pdf